

2. ... (as before).
3. Terminate with answer p .

This works quite well, although there is still a small risk of choosing the least value by chance (or the second-least value, which is almost as bad).

A still better idea is to make the pivot the median of three values taken from the left, middle, and right of the array:

1. Let $pivot$ be the median of $a[left]$, $a[m]$, and $a[right]$ (where m is about midway between $left$ and $right$), swap $pivot$ into $a[left]$ if necessary, and set $p = left$.
2. ... (as before).
3. Terminate with answer p .

This guarantees that the pivot is not the least value, but there is still a small risk of choosing the second-least value by chance.

3.6.5 Comparison

Table 3.45 summarizes the efficiency of the array sorting algorithms we have studied.

The selection sort and insertion sort algorithms both have time complexity $O(n^2)$. However, this simple truth conceals two important points: insertion sort performs only half as many comparisons as selection sort (constant factors do matter when the complexities are the same!), but insertion sort performs far more copies than selection sort. Overall, there is little to choose between them.

The merge-sort and quick-sort algorithms are superior, both having time complexity $O(n \log n)$, at least in the best case. Each has a serious weakness, however. The merge-sort algorithm performs three times as many copies, and needs an auxiliary array of length n , even when the array happens to be sorted. The quick-sort algorithm deteriorates to time complexity $O(n^2)$ in the worst case, which is when the array is sorted or almost sorted and the algorithm is implemented naively. Fortunately, a little care is sufficient to make the worst-case behavior very unlikely to happen in practice, and that makes the quick-sort algorithm a good practical choice.

Table 3.45 Efficiency of array sorting algorithms.

Algorithm	No. of comparisons (approx.)	No. of copies (approx.)	Time complexity	Space complexity
Selection sort	$n^2/2$	$2n$	$O(n^2)$	$O(1)$
Insertion sort	$n^2/4$	$n^2/4$	$O(n^2)$	$O(1)$
Merge-sort	$n \log_2 n$	$2n \log_2 n$	$O(n \log n)$	$O(n)$
Quick-sort (best case)	$n \log_2 n$	$(2n/3) \log_2 n$	$O(n \log n)$	$O(\log n)$
Quick-sort (worst case)	$n^2/2$	0	$O(n^2)$	$O(n)$